INVESTIGATING THE EFFECT OF MIXING TWO SKINNING ALGORITHMS ON AREA PRESERVATION

An Undergraduate Research Scholars Thesis

by

FERAS ABDELLATIF KHEMAKHEM

Submitted to the Undergraduate Research Scholars Program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Shinjiro Sueda

May 2020

Major: Computer Science

TABLE OF CONTENTS

ABSTRACT

Investigating the Effect of Mixing Two Skinning Algorithms on Area Preservation

Feras Abdellatif Khemakhem
Department of Computer Science and Computer Engineering
Texas A&M University

Research Advisor: Dr. Shinjiro Sueda
Department of Computer Science and Computer Engineering
Texas A&M University

We propose to investigate the effects of by mixing weights between two vertex skinning algorithms. The vertex skinning algorithms in discussion are Linear Blend Skinning (LBS) and Dual Quaternion Skinning (DQS), each of which is not perfect if used in isolation. With our proposed algorithm, a per-vertex mixing weight determines the linear combination with which LBS and DQS are employed to find a deformed mesh that is neither lacking in volume due to LBS, nor contains volume gain from DQS. Finding the best mixing weight will allow for more plausible deformations, with respect to shape and volume of models. We will investigate previous attempts to fix the flaws of LBS and DQS , which is important especially when many models are being skinned.

This work extends the work by Yin [1], which was published in 2019, meaning this is still a very novel approach to skinning, without much other published material building off it yet.

The expected outcome is an insight into the validity of Yin's approach in specific setups and how DQS and LBS can be combined at a per-vertex level to sustain mesh area and shape.

CHAPTER I

INTRODUCTION

Skinning in the field of animation is what brings a mesh to life; after mapping bones to different areas of a mesh, skinning enables the mesh to move with the movement of those bones.

1.1 Background and Previous Work

Previous work has found numerous techniques to achieve the goal of skinning, two popular methods being Linear Blend Skinning and Dual Quaternion Skinning. These two methods have relatively low computational cost and work well in an acceptable range of situations [1]. Where these skinning algorithms fail is at bends in the mesh. Linear Blend Skinning (LBS) has an issue of losing volume at bends, known as to yield a collapsing skin effect (see Figure 1 for reference) [2]. This is due to the way LBS uses of linear combinations of matrices to blend transformations. With LBS, transformations are represented as matrices, with each different sections of the matrix representing translation, rotation, and scale. The interpolation of matrices does not preserve the orthogonality of the rotational part of the matrix, inevitably resulting in this undesired effect [2].

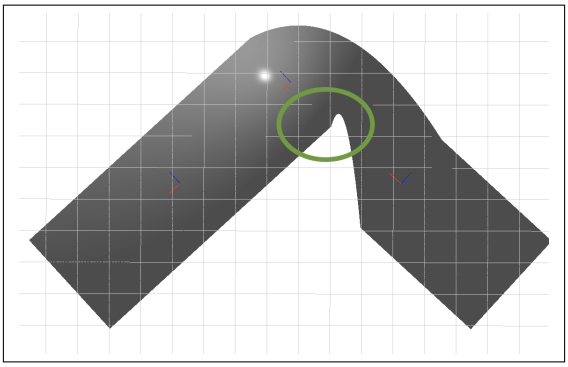

Figure 1: Mesh Skinned by LBS with a Collapsing Joint, Part of Which is Shown in Green

Dual Quaternion Skinning (DQS) has a similar issue, but with a different extreme. DQS has a "bulging joint" effect that results from the same scenario as the collapsing joint effect of LBS [3]. DQS blends rotations in non-linear fashion through the use of dual quaternions in intermediate calculations, avoiding the limitation of LBS, but the bends in the mesh cause the mesh to have an excess of volume, as seen in Figure 2 [4].

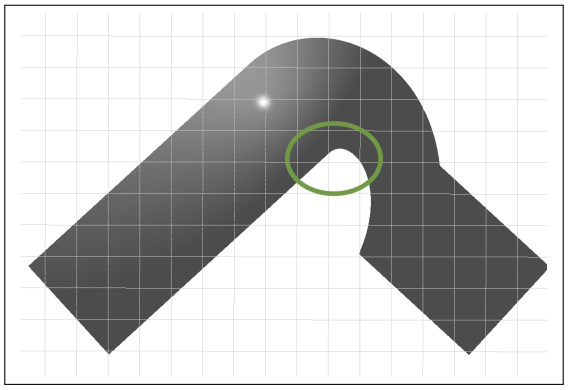

Figure 2: Mesh Skinned by DQS with a Bulging Joint, Part of Which is Shown in Green

More recent work has found a way of combining these two skinning algorithms to create a cost-effective skinning solution that overcomes these limitations [1]. Yin discusses a method of linearly combining LBS and DQS to create a bend with no bulging and no collapsing using a mixing weight, or value between 0 and 1 that combines the resulting matrices of the LBS and DQS. They leave it to future work to propose a scheme for distributing mixing weights, and I hope to do that as a secondary aim of my thesis (mentioned in 1.2). More important than this is finding the actual mixing weights needed to get the most out of Yin's method, which leads me to my aim of this thesis.

1.2 Aims of This Thesis

The main aim of this thesis is to investigate different weights for each vertex in a mesh to prevent loss or gain of volume at bends in a mesh, while still keeping computational cost relatively low. Due to the complexity of this issue, the scope of this thesis will remain in two dimensions, leaving future work to extend to the third dimension. A smaller-scale aim that must be solved first is determining how to distribute weights amongst vertices. Should each vertex get its own weight, or should each weight be assigned to a section of the mesh? How would we divide those sections? These are questions that must be asked in our investigation.

CHAPTER II

METHODS

In order to approach testing our theory, we have developed a simulation program. Due to the uniqueness and need for customability, third party or open source simulations were not flexible enough for our needs - this lead to a need for us to make our own software. The simulation software uses the skinning methods we test and the combinations of those skinning methods. We are mainly concerned with finding the optimal weights to preserve area *and* preserving the shape of the bend. Consequently, analyzing data will be quantitatively collected by recording the area as bend angles increase, and will be qualitatively collected with a visual comparison of the bends.

Our simulation, written in C++ and using programming interface OpenGL, will be an all-in-one solution that will allow us to record our multiple types of data. Due to constant changes in our approach, the program has experienced multiple iterations and improvements throughout the course of our research process, which we will also discuss.

2.1 LBS and DQS Skinning Program

The first iteration of our simulation program simulates LBS and DQS (independently) on a bend in a rectangular mesh. We used C++ for mathematical calculations and graphical setup, and the programming interface OpenGL for visualizing our graphical simulation. The program allowed freedom to create bends at angles from 0 to 180 degrees, in 5 degree intervals. There were many steps toward this iteration and many obstacles were faced in the development of this program, primarily with the influence distribution of bones on each vertex, and the overlapping of vertices.

2.1.1 Developing a Mesh and Rigging It

The first step to bend a mesh was the make a mesh. We created a 2D rectangular using

recursive programming. The number of vertices in the mesh and the dimensions of the mesh depend on user input, and an orthogonal view of the mesh is rendered using OpenGL. Rigging was the next immediate step. Since the number of bones was to also be a possible user input for the simulation, and the dimensions of the rectangle could also change by user input, rigging was not so simple as to be hard-coded. However, all bones were to all be on the z=0 line, meaning the height of the rectangle would not influence the locations of bones.

The bones were placed equidistant from their neighbors, as well as edge bones taking the same distance from the edges of the rectangle. This design choice was made because of the nature of our mesh and the fact that we are trying to isolate one bone per bend, so spatially condensing bones is unnecessary. Bone locations in the mesh were dependent on the width of the mesh and the total number bones. Additionally, bones were organized in a hierarchy so that the bone 0 is a parent to bone 1, bone 1 is a parent to bone 2, et cetera. Figure 3 shows the result of this phase.

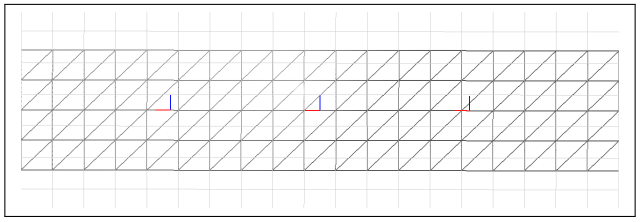

Figure 3: Wireframe of a Mesh Of 20 By 5 Units Rigged With 3 Bones

2.1.2 Rotating the Bones

In order to form a bend on our rectangular mesh, we had to bend some bones. Due to the bones being set up in a hierarchical structure, all rotations made by children bones were based on the rotations of previous bones. Bones were divided into 3 different categories for this simulation. The first category was the parent bone, which was the first bone in every simulation. This bone acted as an angle setter for the first rotation in the bone. The parent bone assumed the angle of one

side of the first . The second category was the bending bone, which categorized all bones at bends. These bends had the most extreme rotations and were at the middle of bends. They were placed at angles opposing the previous bone to make bends from the difference in angle. For example, a 90 degree angle in a bend would come from a 90 degree difference in angles between a parent bone and a bending bone, or two bending bones. The final category of bones was the child bone. This bone was the last bone in every mesh, and assumed the same angle as its parent, effectively acting as an extension of the previous bone. This allowed for better fullness and continuity in the mesh's bends, making it easier to visually analyze bends. These bend types are shown in Figure 4.

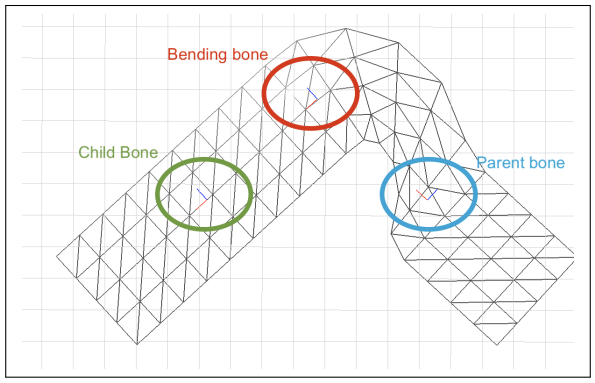

Figure 4: Types of Bones Displayed on a 3-Bone Mesh

2.1.3 Using LBS and DQS

The final component of our first iteration was the use of LBS and DQS to skin, or to move the vertices in accordance with the bones they are mapped to. Both LBS and DQS were programmed using C++. Due to their popularity, many resources provided sufficient information on the details of LBS, meaning only the task of implementation remained. DQS required a deep understanding of dual quaternions and how they can be used to represent transformations. Due to the original program working with transformation matrices, an additional step of transforming the rotation into quaternion form was required before calculating with DQS. Algorithms 1 and 2 show

the structures of LBS and DQS for further context.

Algorithm 1: Linear Blend Skinning

Input: vertex: v; vertex v's bind location: v^0

bind matrix for bone j: B

distance from bone j to vertex v: $dist_{jv}$

transformation matrix applied to bone j: T_j

Output: vertex v's new location: v_f

1 $w_{jv} = \frac{\frac{1}{dist_{jv}}}{\sum_j \frac{1}{dist_{jv}}}$

2 $v_f = \sum_j w_{jv} v^0 T_j B_j^{-1}$

Note that the bind matrix B and bind location v^0 represent matrix and location at the rest position for the mesh. In our case, this rest position is when the angle is 0 and the rectangle is not deformed at all.

Algorithm 2: Dual Quaternion Skinning

Input: vertex: v; vertex v's bind location: v^0

unit dual quaternion based on skinning transformation: $\hat{q}$

distance from bone j to vertex v: $dist_{jv}$

Output: vertex v's new location: v_f

1 $w_{jv} = \frac{\frac{1}{dist_{jv}}}{\sum_j \frac{1}{dist_{jv}}}$

2 $v_f = \frac{\sum_j w_{jv} \hat{q}_j}{|\sum_j w_{jv} \hat{q}_j|}$

At this point, the program could either show a mesh bending up to 180 degrees per bend, using either LBS or DQS. We were still unable to draw conclusions on mixing the two, and no quantitative data was available.

2.2 LBS-DQS Mixing Simulation Program

Similar to our LBS and DQS skinning simulation program, this next iteration on our simu-

lation uses C++ and OpenGL; however, Python was also added to help collect data. At the end of this iteration, our program was able to collect quantitative and qualitative data that reflects effectiveness of mixing LBS and DQS in different proportions. Additional features were added as we saw the need to customize our approach.

2.2.1 Weight Mixing of LBS and DQS

The most significant difference in this iteration was that LBS and DQS were being used in combination to replace what each skinning method previously did individually. Using a modified version of the approached specified by Yin [1] to fit our already calculated LBS and DQS values, the following algorithm was utilized:

Algorithm 3: LBS-DQS Mixing Weights Algorithm

Input: transformation matrix corresponding to the LBS transformation of vertex v: T
transformation matrix corresponding to the DQS transformation of vertex v: Q
mixing weight for vertex v: w

Output: mixed transformation matrix: M

1 $M = (1 - w)T + wQ$

This algorithm takes a linear combination of the transformation matrices that LBS and DQS would have multiplied with vertex v in isolation, to ideally reach a happy medium between area loss and area gain from these skinning methods, respectively. The resulting mixed matrix M was applied to the vertex and its normal in this iteration. At this point, we can visually test the rectangle with LBS, DQS, and a linear combination applied to all vertices.

2.2.2 Mixing Distribution Across Vertices

One immediate shortcoming of applied the mixed weights algorithm to all vertices was that it was more expensive than using LBS or DQS in isolation. Each vertex had to effectively perform the calculations of both LBS and DQS. Both skinning methods are relatively cheap independently, and one goal is to keep our proposed solution cheap as well, so we started exploring applying our

algorithm only to a "critical segment" of the bending rectangle.

The limitation LBS and DQS had that we were attempting to exploit only occur at bends in a mesh, which means that the far ends of our rectangle, beyond the influence of the bend, would not require a mixing of skinning methods as their have no flaws. Thus, the code was further developed to only apply the mixing weights algorithm to a programmable segment that lied between the two bones surrounding the rotating bone.

Another consideration we had was that the mixing weight might not be uniform across the entire critical segment. Consequently, an additional set of parameters we added to our tests was two values, w_0 and w_1, where the former was the mixing weight at the ends of our critical segment. Figure 5 shows where w_0 and w_1 would have the largest influence if the critical segment spanned from the middle bone to the midpoint between itself and adjacent bones. The red lines represent w_0's largest influence, and the green w_1's largest influence. All mixing weights between w_0 and w_1 would linearly interpolate between those values, and the critical segment's bounds are defined by the red lines.

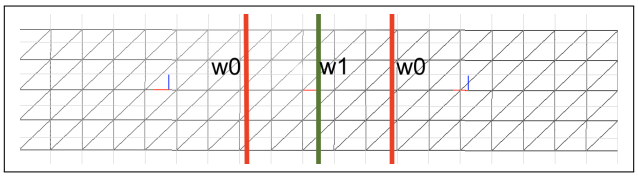

Figure 5: The Locations of Maximum Influence of w_0 and w_1 on a Rectangular Mesh

This model holds some assumptions, the main ones being the following:

1. All vertices sharing a vertical space in the base mesh share the same mixing weight

2. Our bend is symmetrical

Our first assumption entailed that in the original mesh, all vertices that lined up vertically have the same mixing weight. This assumption enables our algorithm to be used in lower complex-

ity, especially in our case, since we only calculated mixing weights once per column of vertices. This meant our calculations saved us an order of $O(c)$ in computational time, where c is the number of vertices in each column. The symmetry of weight distribution about the bending bone was designed with the assumption that the same corresponding vertex on either side of the bend would have the same optimal mixing weight, due to its symmetry. This assumption may limit the usage of this algorithm as the ideal mixing weight on one side could possibly be different than on the other side in an asymmetrical bend, but tests were made with this assumption in mind.

2.2.3 Data Collection

The last phase of our simulation was to encode a testing suite, in order to properly determine which approach and combination of mixing values holds the most success. Our main methods of quantifying data are the following:

- The area of the rectangle for each angle the bending bone takes (quantitative)
- The change in area between angles (quantitative)
- The way a bend looks (qualitative)

The way a bend looks could already be measured in our previous iteration with the use of OpenGL, but the other two methods did not have any support in the program. Our simulation added the ability to calculate the area of the rectangle, by measuring the summation of areas of each triangle made of 3 surrounding vertices. Measuring the area of each triangle allowed for the most precise measure of total area. The angle of the mesh and area of the mesh at that angle were both recorded in a temporary text file. In order to more easily digest the data, a Python solution using the numpy and matplotlib libraries processed the temporary text files, and plotted them on graphs. These graphs also allowed for the change in area between angles to be measured more easily, as this measure was just the slope of a plot between two angles. Finally, a bash script automated the processing of running multiple simulations that recorded areas for multiple angles, with varying values of w_0 and w_1.

CHAPTER III

RESULTS

The simulation software developed in methodology was used to develop a set of qualitative and quantitative tests, in order to determine the efficacy of mixing LBS and DQS in skinning the bend of a rectangle. By testing with multiple values of w_0 and w_1, we can compare our parameters to see how much of an improvement we can create from the original LBS and DQS. The weight in the uniform regions outside of the critical segment, w_u, is also a variable that becomes critical to change as we realize soon. All tests are done with a 3-boned rectangle of 10 units width and 2 units height, where the middle bone is the bending bone. This means the ideal area of our mesh is 20 units. All graphs are made with 1000x100 vertex rectangle simulations, but all figures are 100x10 vertices for easier visual interpretation. Finally, our critical segment's bounds are at the midpoints between the rotating bone and adjacent bones.

3.1 Our Baseline: LBS and DQS

Data collected from the second iteration of our simulation software was used in conjunction with the skinning in our first iteration to create a controlled baseline with which to compare performance in determining whether or not improvements were made when mixing weights. Figures 6 and 7 show our rectangle bending at 90°angle using LBS and DQS, respectively.

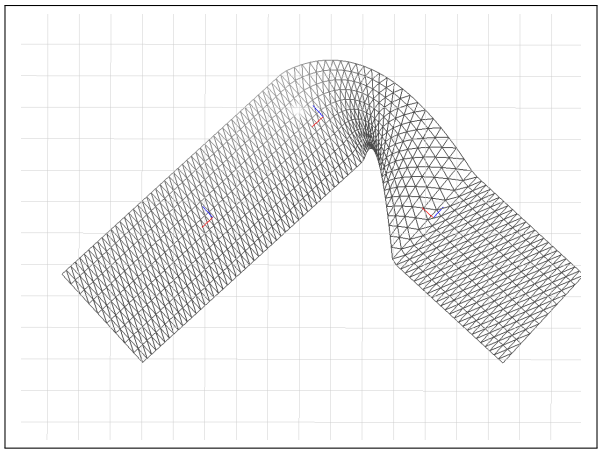

Figure 6: Mesh at 90°Angle Bend Using LBS

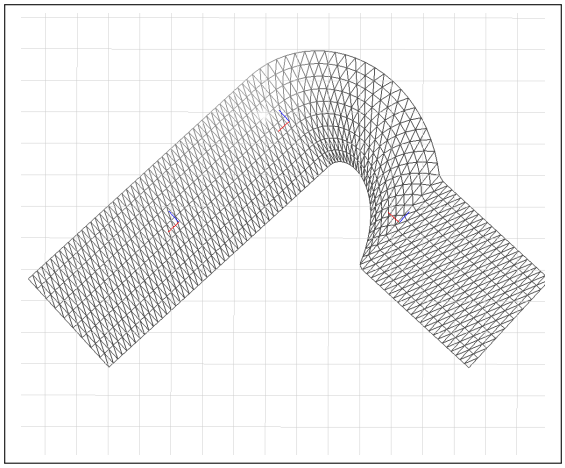

Figure 7: Mesh at 90°Angle Bend Using DQS

The difference between Figures 3 and 4 show us the extremes in the issue of changing area at bends. Figure 6's small curve on the outer end of the bend reflects the collapsing joint issue that is unique to LBS. Figure 7's much larger and rounder outer curve shows the contrasting bulging joint effect that leads to an excess of area. The areas of the rectangles using LBS and DQS are also recorded below, in Figure 8.

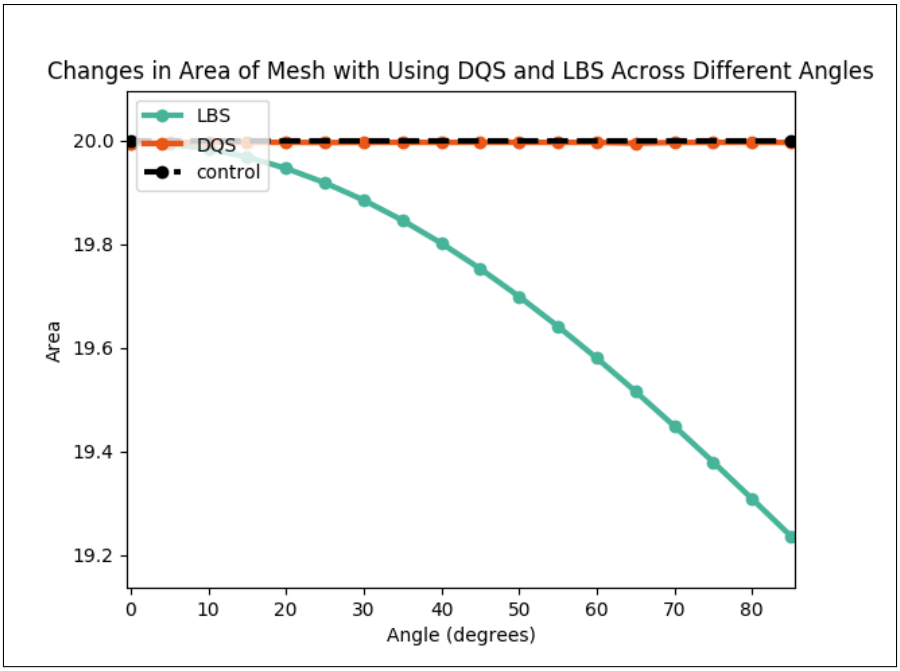

Figure 8: Change in Area of LBS and DQS Over Different Bend Angles

It should be noted that DQS's area does not increase nearly as much as LBS's area decreases, but the visual deformation of DQS is the parameter that requires DQS to be modified.

3.2 Weight Mixing Results

Using our second iteration of our simulation software, data was collected for meshes utilizing the mixing of weights in the critical segment. Different combinations of w_0 and w_1 were each tested in 0.25 increments between 0 and 1. The resulting areas of each combination is reflected in Figure 6 below. Note for Figure 9, the first value of each name in the legend corresponds to the value of w_0, and the second w_1.

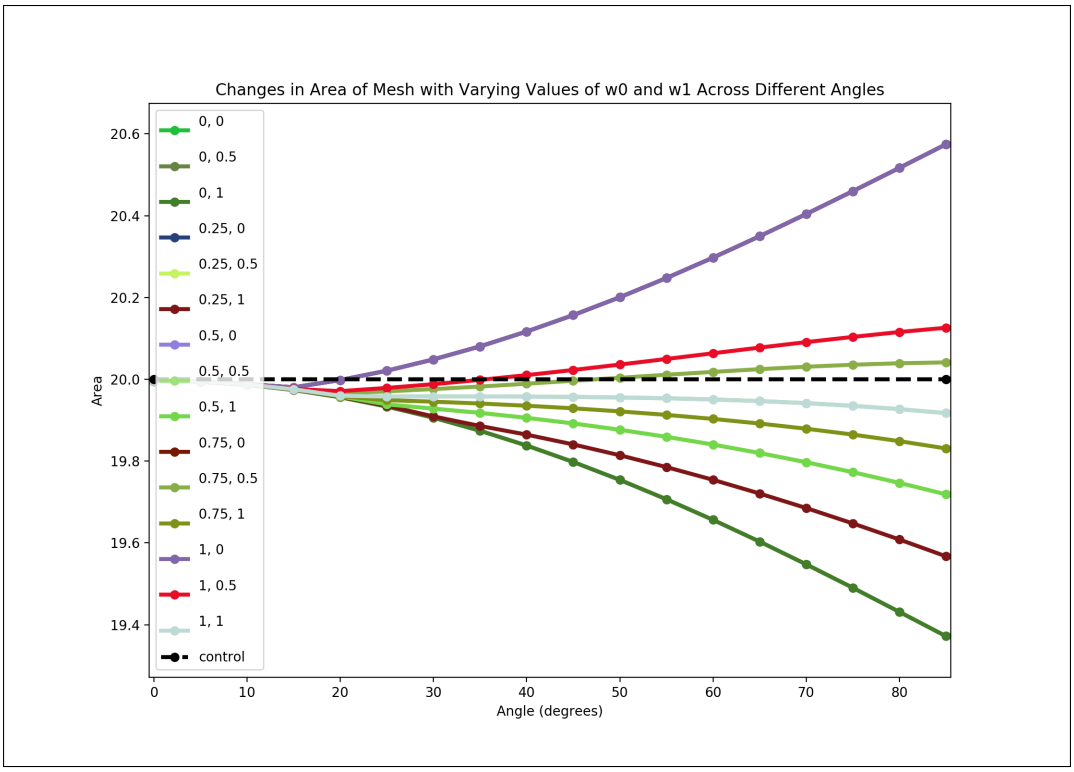

Figure 9: Change in Area of Mesh with Varying w_0 and w_1 Values Over Different Bend Angles and $w_u = 0$

The control line represents a constant area of 20 units, which is the ideal area in all cases. All values of w_u used in Figure 9 were 0, signifying that all skinning outside the critical segment was done with LBS. For reference, $0, 0$ reflects LBS, and $1, 1$ reflects DQS. The values $0.75, 0.5$ have the closest values to 20 at the full angle; however, DQS has the most area preservation over time. This means that w_0 and w_1 of 0.75 and 0.5 works the best when bent at a higher angle, whereas DQS works best in the process of bending a mesh, at the area changes less and thus it looks more natural. Unfortunately, further investigation into the way the meshes look trails us to a problem, as highlighted in the next section.

3.3 Smoothness of Mixing Weights

3.3.1 The Problem

One issue with our data arises when looking at the meshes that result from some of these combinations of w_0 and w_1. By setting the bounds of the critical segment to be the midpoints between adjacent bones and the bending bone, an assumption is made that the deformations caused by DQS and LBS is only limited to the vertices with more influence from the bending bone than other bones.

Although the numbers may look nice after making this assumption, further investigation shows that the assumption was wrong to make. The collapsing and stretching of the mesh in Figure 10 and Figure 11 reflect that this assumption's flaws as seen below, respectively. By setting w_0 and w_1 to 0, Figure 10 works exactly like LBS in the critical segment, and setting the variables 1 to allows Figure 11 to effectively use DQS in the critical segment.

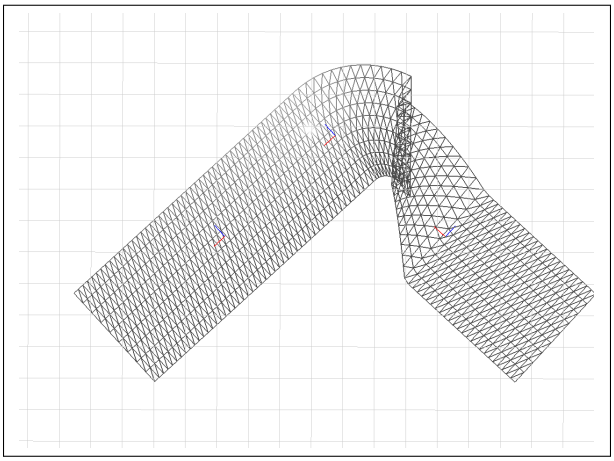

Figure 10: Mesh at 90°Angle Bend With LBS in Critical Segment and DQS in Outer Regions

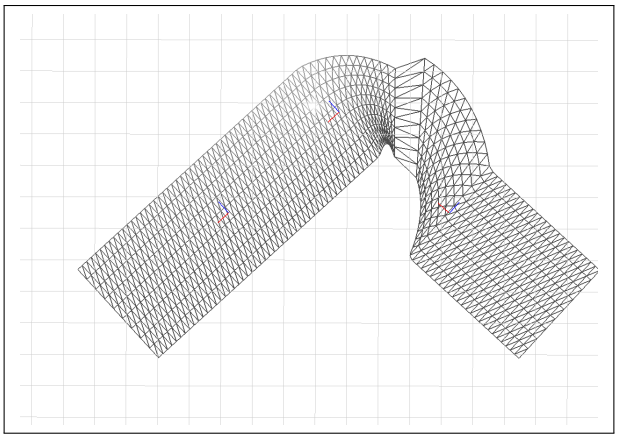

Figure 11: Mesh at 90°Angle Bend With DQS in Critical Segment and LBS in Outer Regions

The overlapping and spreading of the vertices on Figures 10 and 11 are due to a sudden shift between skinning methods in an area where both act differently. The left side of the bending bone (the middle bone) looks completely natural due to the equivalent functionality of LBS and DQS on straight edges; however, the behavior of LBS and DQS differ so much at bends that the right side of the bending bone fails to smoothly transition between skinning methods.

3.3.2 The Solution

Further investigation has shown that "smoothing" the edges of the critical segment will fix this issue. By setting the same mixing weight w_u outside the critical segment to the value w_0 of the mixing weight at the edges of the critical segment, an effective smoothing of the vertices' mixing weight will allow for the behavior at the critical segment's edges to be less abrupt and more seamless. Figure 12 shows an example of smoothing where w_u and w_0 are both 0 and w_1 is 1, meaning the center has DQS behaviour and transitions to an LBS behaviour as it reaches out to surrounding bones.

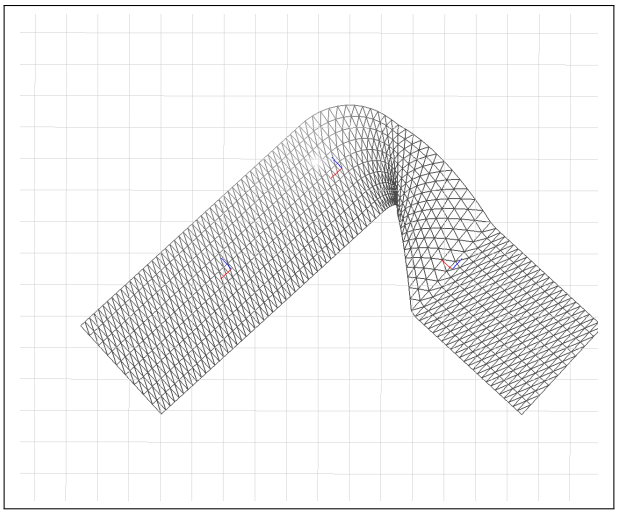

Figure 12: Mesh with $w_u = w_0 = 0$ and $w_1 = 1$

The same tests run earlier were re-run with w_u values equaling w_0, and their results can be see in Figure 13. Further investigation with this smoothing might lead toward a a more natural-looking bend with fully preserved area.

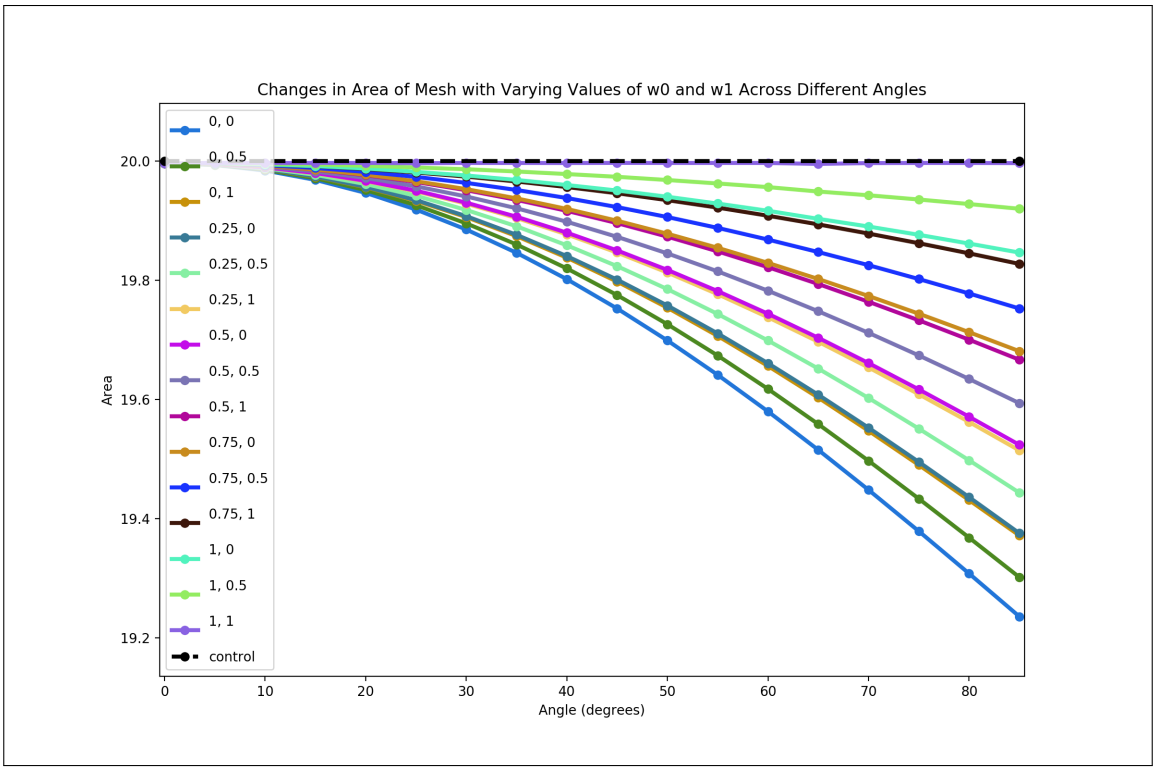

Figure 13: Change in Area of Mesh with Varying w_0 and w_1 Values Over Different Bend Angles, Where $w_u = w_0$

The difference between this graph and the graph presented when $w_u = 1$. This graph's end bounds, which reflect LBS (0, 0) and DQS (1, 1), more accurately reflect the original LBS and DQS graph mentioned in Section 3.1. Although the previous graph in Section 3.2 may "look nicer", the unintentional visual deformation that existed in previous data invalidated any results. Of all of these combinations, DQS has the best area preservation over changing angles and closest area to to the original 20 units at the highest angle. With respect to the qualitative visual accuracy of the bend. All bends in this set of data no longer have collapsed or scratched meshes; however, determining the most "visually realistic" bend is subjective and up to the user.

CHAPTER IV

CONCLUSIONS

4.1 Conclusion

Throughout this investigation, we have successfully developed a simulation software with which different combinations of LBS and DQS may be applied to a 2D mesh. Of the values tested that mix between DQS and LBS, none of them had the optimal effect except for values of 0, meaning DQS has the most success conserving area and keeping it close to the original area. Consequently, the investigation led to some possible methods to mix the weights, but the validity of these methods cannot be verified without further investigation. The current most viable usage of this algorithm is to make bends seem more realistic, at the cost of losing area.

4.2 Limitations

The largest flaw with our evaluation is the way bones are modeled in our simulation. In Yin's work, it was shown that DQS' bulging occurs when blending is done using the frames at the bones [1]. Although our simulation does this too, Yin's model (and a more standard model) consists of long bones that connect at joints; however, our model treats bones at points, meaning the joints between bones consists of a large gap. Additionally, we place a bone at each bend and joints between bends, as opposed to a joint at each bend and a bone at between each bend.

4.3 Future Work

This investigation has formed some paths for future work. The first suggested future work is to further investigate the point between the bending bone and adjacent bones at which the negative effects of bulging joints and collapsing joints are no affected. This will likely depend on the way the influence of bones on shared vertices is distributed or the type of bend the bone experiences, but

our assumption that it would be between two bones was proven incorrect, as reflected in the first set of our results. Determining this point in a bend could allow the use of weight mixing to only have to exist within the critical area, accomplishing the goal of keeping the algorithm efficient.

Additionally, investigating with asymmetrical weight distributions would make for a good next step. Our assumption that the bend would be symmetrical was obviously not true, and although one side of our mesh seemed the same regardless of the mixing weight applied to it, other situations when the both halves of the bone experience unwanted deformations might require an additional weight w_2 for the endpoint opposite to w_0 in the critical segment.

Finally, extending this investigation to determine optimal mixing weights with a mathematical mode would make this approach more valid. Currently there is only slight improvements with most combinations of numbers, as seen in Figure 13 of section 3.3, but developing a method to determine the ideal mixing weight in any situation would allow this algorithm to become an option used in practice.

REFERENCES

[1] H. Yin and R. Mukundan, "Improved vertex skinning algorithm based on dual quaternions," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, IEEE, 2019.

[2] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann, "Joint-dependent local deformations for hand animation and object grasping," in *In Proceedings on Graphics interface'88*, Citeseer, 1988.

[3] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan, "Geometric skinning with approximate dual quaternion blending," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 4, pp. 1–23, 2008.

[4] L. Kavan and J. Žára, "Spherical blend skinning: a real-time deformation of articulated models," in *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pp. 9–16, 2005.